KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma / Ohjelmistotekniikka

Hannu Niemelä

OPTIMOINTIOHJELMAN PÄIVITTÄMINEN JA TUOTEKEHITYS

Opinnäytetyö 2013

TIIVISTELMÄ

KYMENLAAKSON AMMATTIKORKEAKOULU
Tietotekniikka
NIEMELÄ, HANNU

Optimointiohjelman päivittäminen ja tuotekehitys
Opinnäytetyö
29 sivua
Työn ohjaaja
Yliopettaja Paula Posio
Toimeksiantaja
Haminan Energia Oy
Huhtikuu 2013
Avainsanat
.NET Framework, C#, sähköntuotanto, optimointi

CHP -optimoija on Haminan Energia Oy:n käyttämä sähkön ja lämmön yhteistuotantolaitosten toimintaa optimoiva sovellus, joka lukee sähkön pörssihinnat www-sivulta ja muut laskentatiedot tietokannasta. Ohjelma laskee niiden perusteella, milloin voimalaitoksia on taloudellisesti kannattavaa ajaa ja lähettää ajo-ohjeet laitoslogiikalle automaattisesti. Ohjelma on alun perin toteutettu 10 vuotta sitten Visual Basic 6 -ohjelmointikielellä, ja se ei enää toimi uudemmissa käyttöjärjestelmissä. Myös tuotantoympäristössä on tapahtunut muutoksia, jotka vaativat muutoksia ohjelman toiminnallisuuteen.

Opinnäytetyön aiheena oli päivittää sovellus toimimaan Microsoft Windows 7 -käyttöjärjestelmässä ja ohjelmoida tarvittavat muutokset optimointiprosessin toimintaan. Työhön sisältyi myös sovelluksen käyttöliittymän uudistaminen.

Sovellus toteutettiin uudelleen .NET -tekniikalla ja C#-ohjelmointikielellä. Käyttöliittymä toteutettiin Windows Forms -komponenteilla.

Lopputuloksena saatiin toimiva sovellus, joka täyttää sille asetetut vaatimukset. Siihen oltiin yrityksessä tyytyväisiä ja se otettiin käyttöön.

ABSTRACT

KYMENLAAKSON AMMATTIKORKEAKOULU
University of Applied Sciences
Information Technology

NIEMELÄ, HANNU	Update and product development of optimization software
Bachelor's Thesis	29 pages
Supervisor	Paula Posio, Senior Lecturer
Commissioned by	Haminan Energia Oy
March 2013	
Keywords	.NET Framework, C#, power production, optimization

CHP – Optimoija is optimization software used by Haminan Energia Oy. The software searches electricity prices from power market's webpage and other data from database. Based on this information it calculates whether it is economically more advantageous to operate plant or to buy power from the market. The software was originally programmed with Visual Basic 6 - programming language and the software has become obsolete for modern operation systems. There have also been changes in the production environment which requires modifications in the functionality of the software.

The objective of this study was to re-program the software to run in Microsoft Windows 7 operating system and implement the desired modifications to the optimization process. The objective also included the design and implementation of a new user interface.

The work was carried out with .NET framework technology and C# programming language. The new user interface was created by using Windows Forms components.

As a result, new software was created. It fulfilled the requirements given and it was successfully deployed in the company.

SISÄLLYS

TERMIT JA LYHENTEET

CHP	Lyhenne sanoista Cogeneration of Heat and Power, sähkön ja lämmön yhteistuotanto
.NET Framework	Microsoftin kehittämä ohjelmistokomponenttikirjasto, joka tarjoaa käyttöjärjestelmän päällä toimivan ajonaikaisenympäristön
InSQL	Wonderware Industrial SQL Historian, teollisuuslaitoksille suunniteltu tietokanta ja tiedon keruujärjestelmä.
Visual Basic 6	Microsoftin luoma BASIC–sukuinen ohjelmointikieli.
C#	Microsoftin .NET Framework:a varten kehitetty ohjelmointikieli
Windows Forms	.NET Framework:lle suunniteltu käyttöliittymäkomponenttikirjasto

1 JOHDANTO

Haminan Energia Oy:n käytössä on diplomityönä 10 vuotta sitten toteutettu automaattinen optimointijärjestelmä pienille sähkön- ja lämmöntuotantolaitoksille (CHP-laitos). Sillä pyritään varmistamaan yhteistuotantolaitosten mahdollisimman taloudellinen ajotapa muuttuvissa markkinaolosuhteissa. Markkinaolosuhteet muodostuvat sähkön hinnoista sähköpörssissä, kaasun hankintahinnasta ja kohteen paikallisista sähkö- ja lämpökuormituksista. Optimointijärjestelmä ottaa myös muita laitoksen talouteen vaikuttavia tekijöitä huomioon. Periaatteena on, että laitoksia ajetaan vain silloin, kun sähkön tuottaminen tulee halvemmaksi kuin sen ostaminen sähköpörssistä. Tällä hetkellä järjestelmään on kytketty kaksi voimalaitosta, mutta järjestelmään on myös mahdollista lisätä uusia laitoksia

Sähkön pörssihinta muodostuu asiakkaiden jättämien osto- ja myyntitarjousten perusteella. Asiakkaat jättävät osto- ja myyntitarjouksensa seuraavan päivän jokaiselle tunnille ja kaupankäyntijärjestelmä laskee seuraavan vuorokauden pörssihinnat.(1,17.)

Automaattiseen optimointijärjestelmään kuuluvia osia ovat laitoslogiikka, laitoslogiikan lukuosasovellus, tietokanta, CHP -optimoijasovellus ja hinnanlukijasovellus. Kaksi jälkimmäistä oli toteutettu Visual Basic 6 – ohjelmointikielellä ja ne eivät enää olisi toimineet uudemmissa käyttöjärjestelmissä. Lisäksi laitosympäristössä oli tapahtunut muutoksia, jotka vaativat sovelluksen toiminnallisuuteen muutoksia. Yrityksellä oli siis tarve uudistaa järjestelmää, jotta vanhentuneesta käyttöjärjestelmästä päästäisiin eroon ja optimointisovellus saataisiin ajan tasalle.

Opinnäytetyön tarkoitus on toteuttaa optimointisovellus uudelleen toimimaan nykyaikaisessa käyttöjärjestelmässä ja toteuttaa halutut muutokset optimoinnin toimintaan. Uudelleentoteutus sisältää myös uuden käyttöliittymän suunnittelun. Pyrkimyksenä on myös tehdä lähdekoodista modulaarisempaa.

Työ päätettiin opinnäytetyön ohjaajan neuvosta toteuttaa .NET-tekniikalla Windows Forms -komponentteja käyttäen. Kieleksi valittiin C# ja kehitystyökaluksi Visual Studio 2012. Kyseisistä tekniikoista oli vähän kokemusta jo aikaisemmista kouluprojekteista ja kursseista, mutta uutta opittavaa riitti silti runsaasti.

Työ tehtiin suurelta osin Haminan Energia Oy:n toimistotiloissa kesällä 2012. Haminan Energia Oy on Haminan kaupungin omistama energiayhtiö. Sen tärkeimmät toimialat ovat sähkön ja maakaasuun myynti ja jakelu, mutta sen toiminta ulottuu myös kaukolämmön jakeluun ja tiedonsiirtopalveluihin. Se työllistää vakituisesti noin 40 työntekijää ja sen vuotuinen liikevaihto on yli 30 miljoonaa euroa.

2 JÄRJESTELMÄN KUVAUS

2.1 Optimointialgoritmi

Voimalaitoksen tulot koostuvat sähkön ja lämmön myynnistä asiakaskohteeseen ja sähkön myynnistä yrityksen taseeseen. Menot koostuvat pääosin maakaasun ostosta ja huoltokustannuksista. Optimointialgoritmi laskee tulojen ja menojen perusteella milloin laitosta on kannattavaa ajaa. (1, 34 - 35.)

2.2 Käyttötavat

Optimointisovellus laskee ajo-ohjeita seuraavalle minuutille, tunnille ja vuorokaudelle. Optimointijärjestelmä tukee kahta eri käyttötapaa:

- ”teho-ohje”-käyttö

  Teho-ohjekäytössä voimalan generaattorille annetaan suoraan sähköteho-ohje prosenttilukuna 0 – 100.

- ”ajomalli”-käyttö

  Ajomallikäytössä generaattorille annetaan yksi neljästä mahdollisesta ajomallista. Laitoslogiikka osaa itse säädellä sähkögeneraattorin tehoa ajomallin perusteella. (1, 30 - 32.)

2.3 Voimalaitokset

Optimointijärjestelmä ohjaa tällä hetkellä kahta asiakaskohteissa olevaa kaasumoottorivoimalaa. Voimalat ovat nimeltään Harju ja BASF. Kohteessa Harju on yksi 770

kW:n sähkötehon omaava aggregaatti ja kohteessa BASF niitä on kaksi. Voimalat ovat käytännössä varavoimakäytössä, mutta niitä on järkevää ajaa myös silloin, kun se on taloudellisesti kannattavaa. (1,28.)

2.4 Microsoft SQL server ja Wonderware Industrial SQL Server Historian

Microsoftin SQL Server on tietokantapalvelin, joka toimii Windows – ympäristössä.

Industrial SQL Server toimii siltana teollisuuslaitoksen reaaliaikaisen hallinta- ja valvontaympäristön ja joustavan liiketoimintaympäristön välillä. Sen käyttö on vahvasti kytköksissä Microsoftin SQL Serveriin. Industrial SQL Server Historian tallentaa tietoa teollisuuslaitosten toiminnasta. Tietoihin pääsee käsiksi OLE DB – yhteyden avulla.(2,3)

InSQL – palvelin on optimointijärjestelmän tärkeä tietolähde, josta saadaan reaaliaikaista dataa voimalaitosten tilasta. InSQL :n dataan pääsee käsiksi Microsoftin SQL Serverin kautta.

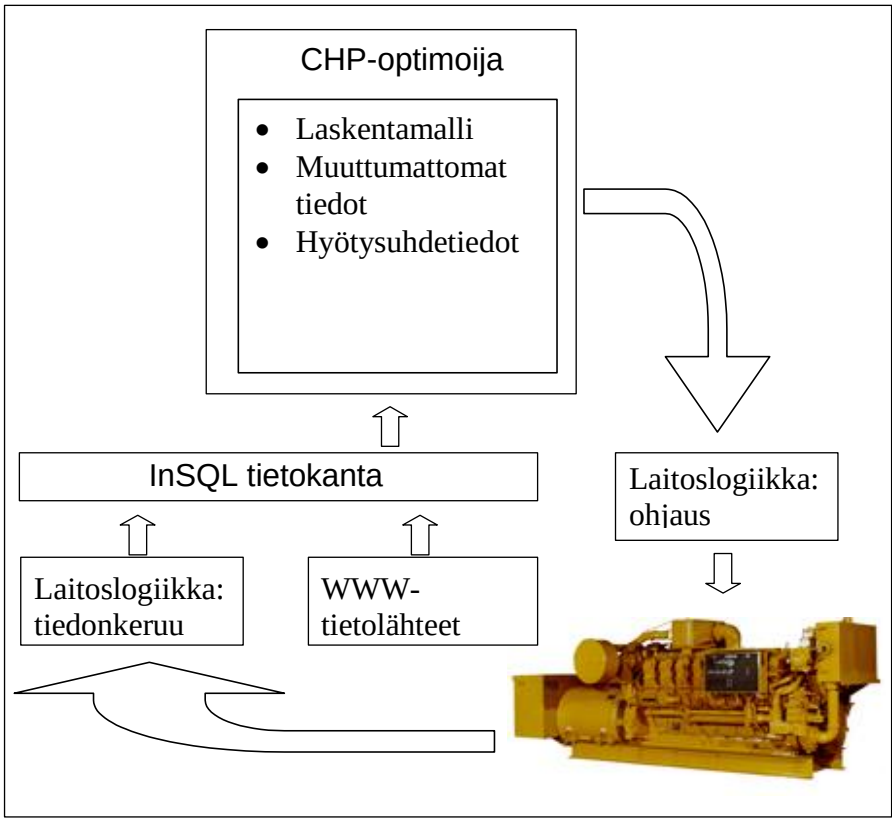
2.5 Optimointijärjestelmän toimintaperiaate

Kuvassa 1 näkyy optimointijärjestelmän toimintaperiaate. Pääohjelma on CHP -optimoija. Se hakee laskemiseen tarvittavat tiedot tietokannasta ja laskee optimiajotavat laitoksille. Alkuperäisessä sovelluksessa pystyy valitsemaan joko teho-ohjekäytön tai ajomallikäytön, sekä reaaliaika- tunti- tai vuorokausioptimoinnin.

Hinnanlukija on sovellus, joka hakee sähkön pörssihinnat www-sivulta ja tallentaa ne tietokantaan.

Tietokantana on Microsoft SQL Server 2000, johon tallennetaan hintatiedot ja staattiset laskentatiedot. SQL Serverille integroitu InSQL- palvelin kerää reaaliaikaista ja tietoa voimalaitoksien toiminnasta ja tilasta. Järjestelmä käyttää myös kahta Access -tietokantaa ajo-ohjeiden tallentamiseen.

Laitoslogiikan lukuosasovellus lukee CHP -optimoijan laskemat ajotavat Access-tietokannasta ja siirtää tiedot laitoslogiikalle, joka ohjaa laitoksen toimintaa.



Kuva 1. Järjestelmän toimintaperiaate

3 TEKNOLOGIA

3.1 .NET Framework

.NET Framework on Microsoftin kehittämä ohjelmistokomponenttikirjasto. Se tarjoaa johdonmukaisen oliosuuntautuneen ohjelmistokehitysympäristön ja koodin suoritusympäristön.

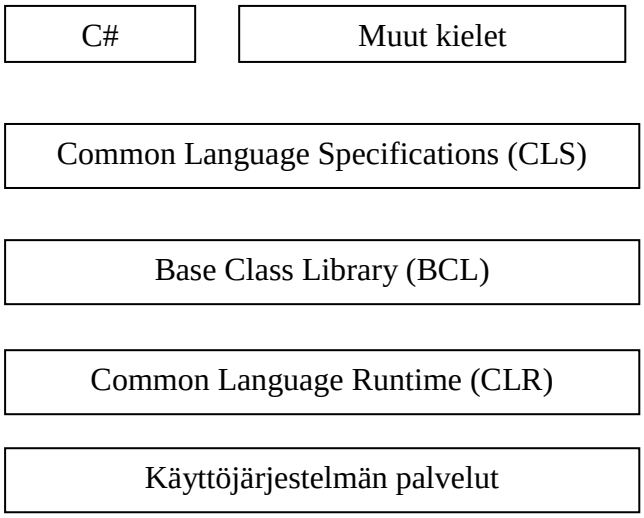
.NET Framework:n tärkeimmät osat (kuva1.)

Common Language Specification (CLS) sisältää määritykset .NET -kielelle. Esimerkiksi muuttujatyyppirakenteen on kaikissa .NET - kielissä oltava samanlainen.

Base Class Library (BCL) on kaikille .NET -kielille yhteinen perusluokkakirjasto.

Common Language Runtime (CLR) on käyttöjärjestelmän päällä toimiva ajonaikainen ympäristö. Se ohjaa käskyjen ja säikeiden suorittamista, hallitsee muistinkäsitte-

lyä ja tarjoaa erilaisia palveluita helpottamaan ohjelmointia. Tärkeimpiä CLR:n tarjoamia etuja on muistinhallintaa helpottava automaattinen roskienkeruu. (4,9.)(13.)



Kuva 2. .NET Framework rakenne

3.2 C#

C# on Microsoftin kehittämä suosittu nykyaikainen oliosuuntautunut ohjelmointikieli, joka on laajalti yrityskäytössä. Se julkaistiin vuonna 2000 ja se on suunniteltu erityisesti Microsoftin .NET –framework:ssa toimivien sovellusten tekemiseen. C# on kehitetty C- ja C++-kielistä ja muistuttaa syntaksiltaan paljon Java-kieltä.

Vaikka C#: ei periaatteessa ole sidottu ainoastaan .NET Framework:iin, on C# kuitenkin käytännössä nimenomaan .NET – Framework:n kieli.(4,9.)

3.3 Windows Forms

Windows Forms on .NET Framework:lle suunniteltu ohjelmointirajapinta ja käyttöliittymäkomponenttikirjasto, joka yksinkertaistaa monia yleisiä ohjelmointitehtäviä, kuten tiedostonkäsittelyä.

Windows Formsissa lomake (Form) on visuaalinen ikkuna, joka näyttää käyttäjälle informaatiota. Sovelluksia luodaan laittamalla lomakkeelle kontrolleja (controls), jotka reagoivat käyttäjän toimenpiteisiin, esimerkiksi hiiren ja näppäinten painalluksiin. Reagointi tapahtuu tapahtumankäsittelijöiden (eventhandler) avulla.

Windows Forms sisältää monia valmiita kontrolleja. Niitä ovat muun muassa painonappi, radionappi, tekstikenttä tai jopa websivu. On myös mahdollista luoda omia komponentteja.(5.)(14.)

Tässä opinnäytetyössä yleisimmin käytetyt käyttöliittymäkomponentit on esitetty taulukossa 1:

Taulukko 1. Yleisesti käytettyjä Windows Forms –komponentteja

Kontrollin nimi	**Kontrollin funktio**
Label	Näyttää tekstiä
Button	Hiirellä klikattava painonappi
TextBox	Tekstikenttä, johon käyttäjä voi kirjoittaa
MainMenu	Formille liitetävä valikko
CheckBox	Sisältää boolean arvon. Arvo on tosi, jos kenttä on merkitty.
RadioButton	Sisältää boolean arvon.
GroupBox	Ryhmittää kontrolleja samaan ryhmään

Kontrolleja voi asettaa lomakkeelle joko Visual Studion graafisen työkalun avulla tai ohjelmallisesti. Tässä opinnäytetyössä kontrollit luodaan ohjelmallisesti. Ohjelmallinen eli dynaaminen komponenttien luominen mahdollistaa käyttöliittymän, joka on joustavampi muutoksille.

Ohjelmallisesti kontrolli luodaan kuten kuvassa 3. Ensin luodaan olio, joka on kyseisen kontrollin typpinen. Sitten kyseinen olio lisätään esimerkiksi lomakkeen kontrollien joukkoon. Jos luodun kontrollin pitää reagoida johonkin käyttäjän toimenpiteisiin,

kuten hiiren klikkaukseen, täytyy komponenttiin lisätä tapahtumankäsittelijä. Lisäksi on toteutettava metodi, joka käsittelee tapahtuman.(6.)(7.)

```
Label myLabel1 = new Label();
myLabel1.Text = "esimerkki";
myLabel1.Location = new Point(10, 10);
this.Controls.Add(myLabel1);


void myLabel1_Click(object sender, EventArgs e)
{
    Label myLabel1 = (Label)sender;
    myLabel1.Text = "clicked";
}
```

kuva 3. Label - kontrollin luominen ohjelmallisesti lomakkeelle.

3.4 OLE DB

OLE DB on kokoelma Component Object Model (COM) – ohjelmointirajapintoja. Se tarjoaa sovelluksille yhdenmukaisen pääsyn erilaisiin tietolähteisiin sekä tarjoaa mahdollisuuden toteuttaa lisäpalveluja tietokantoihin.(8.)

OLE DB noudattaa johdonmukaisesti yleismallia siitä, miten sovellukset pääsevät käsiksi dataan. OLE DB olettaa, ettei sovelluksella ole suoraa pääsyä dataan, vaan data sijaitsee erillisessä tallennuspaikassa, kuten tiedostossa, sähköpostissa tai tietokannassa. Sovellus pyytää dataa välikädeltä, joka palauttaa kopion kysytystä datasta sovellukselle. Sovellukselle toimitettu data pidetään välimuistissa, mikä helpottaa sen käsittelyä ja muokkaamista. Tässä mallissa sovellus on kuluttaja ja välikäsi on tuottaja. Kuluttaja-tuottaja malli (concept of consumer and provider) on OLE DB:n perusta, ja OLE DB -teknologiaa tulisi aina ajatella sen pohjalta.(9.)

3.5 HTML Agility Pack

HTML Agility Pack on ilmainen .NET- koodia käyttävä HTML-parseri. Se osaa lukea HTML- dokumentin ja muodostaa siitä puumaisen tietorakenteen, jota voi käsitellä ohjelmallisesti(10).

3.6 Visual Studio

Kehitystyökaluna käytettiin Visual Studio 2012 Professional Versiota. Visual Studio on Microsoftin suosittu ohjelmistokehitystyökalu, jolla voi kehittää ohjelmia monenlaisiin ympäristöihin useilla eri kielillä.(11.)

Windows Forms projektin luominen onnistuu Visual Studiossa automaattisen prosessin avulla(12).

4 TYÖN VAIHEET

Optimointijärjestelmän uudistaminen aloitettiin perehtymällä vanhan sovelluksen toimintaan tutkimalla sen lähdekoodia, käyttöohjetta ja diplomityödokumenttia, sekä haastattelemalla työntekijöitä, jotka ovat sovellusta käyttäneet. Tarkoituksena oli saada selville sovelluksen toiminta ja uudistetun version toimintavaatimukset.

Haastetta aiheutti dokumentaation vähäisyys ja käyttöohjeen tulkitseminen. Sovellukseen oli tehty joitakin muutoksia käyttöohjeen kirjoittamisen jälkeen, eikä niitä ollut päivitetty käyttöohjeeseen. Visual Basicin syntaksi ei ollut entuudestaan täysin vierasta, mikä helpotti vanhan ohjelman lähdekoodin lukemista. Lähdekoodin tutkimista hankaloitti kuitenkin erityisesti kommenttien puute ja testiajomahdollisuuden puute. Myöhemmin tuli myös ilmi tilanteita, joissa esimerkiksi funktion nimi oli harhaanjohtava, mikä aiheutti hämmennystä. Työn aihepiiri ei ollut entuudestaan tuttu, joten asioiden omaksuminenkin vei aikansa.

Työ suoritettiin Haminan Energia Oy:n toimistotiloissa. Työn tekeminen oli hyvin itsenäistä, ja ainoa apu varsinaiseen ohjelmointiin oli saatavilla internetsivuilta. Muihin teknisiin ongelmiin, sekä tietenkin ohjelmiston vaatimuksiin ja toiminnallisuuden suunnitteluun, sai apua ja neuvoja muilta yrityksen työntekijöiltä kiitettävästi.

4.1 Ohjelmistokehitysmenetelmä

Ohjelmistokehitysmenetelmänä oli käytössä evoluutioprototyyppimenetelmä. Evoluutioprototyypillä tarkoitetaan menetelmää, jossa ohjelmiston prototyyppiversiosta kehitetään vaiheittain lopullinen versio. Evoluutioprototyyppi muistuttaa hyvin paljon iteratiivista kehitysmenetelmää.

Prototyyppimenetelmässä ohjelman komponenteista tehdään ensin kokeiluversio, jotta komponentin toimintaa voidaan testata. Myöhemmin prototyypin koodi voidaan korjata todellisella tuotantokoodilla. Usein prototyyppien koodia voi käyttää uudelleen tuotantokoodissa.

Menetelmän ongelmana on houkutus jättää epätäydellisesti toimiva prototyyppikomponentti lopulliseen versioon, varsinkin projektin loppuvaiheessa, kun aikaa on vähän. Myös prototyypeissä piilevät ongelmat saattavat jäädä huomaamatta testauksessa ja tulevat esiin vasta lopullisessa versiossa.(13,38 – 39.)

4.2 Tietokantayhteydet

Tietokantayhteyksien käsittely hoidettiin OLE DB -ohjelmointirajapinnan kautta. Käsittelyn helpottamiseksi luotiin Tietokanta-luokka, joka vastaa yhteyden avaamisesta ja sulkemisesta tietokantaan. Käytännössä ohjelman kaikki SELECT - ja DELETE - kyselyt suoritetaan tämän luokan jäsenmetodien avulla. INSERT - tyyppiset kyselyt oli helpompaa sijoittaa suoraan koodiin.

Suurin osa SQL – komennoista oli sellaisenaan kopioitavissa alkuperäisen sovelluksen lähdekoodista, mutta joitakin komentoja täytyi laatia itse.

4.3 Hintojen hakeminen www-sivulta

Alkuperäisessä järjestelmässä hintojen hakemisen suorittava sovellus oli erillään varsinaisesta optimointisovelluksesta, koska sen oli toteuttanut eri henkilö kuin muun sovelluksen. Yksinkertaistamisen ja käytön helpottamisen vuoksi hintojen hakeminen toteutettiin uudistetussa sovelluksessa samaan sovellukseen optimoinnin kanssa. Hintojen haun tehtävä on hakea kerran päivässä seuraavan vuorokauden sähkön pörssihinnat www-sivulta ja tallentaa ne tietokantapalvelimelle.

Vanhassa versiossa käytetty ohjelmistokirjasto websivun lataamiseen ja läpikäymiseen ei ole yhteensopiva .NET Framework:n kanssa, joten työssä päädyttiin käyttämään Html Agility Pack -kirjastoa. Se osoittautui helppokäyttöiseksi ja sopivan kevyeksi tähän kyseiseen tehtävään.

Hintojen haku perustuu HTML-dokumentin elementtien iteroimiseen ja indeksoimiseen Html Agility Packin luomasta puurakenteesta. Hintojen hakemisen suorittava funktio etsii Suomen pörssihintojen sarakkeen html -taulukosta ja tallentaa hinnat tietokantaan. Funktio pystyy ottamaan huomioon mahdolliset pienet websivuun tehtävät muutokset, kuten sarakkeen lisäämisen taulukkoon tai Suomen hintojen paikan vaihtumisen. Mutta jos sivun rakennetta muutetaan radikaalisti, ei funktio enää toimi. Tämä sama ongelma oli myös vanhassa versiossa, eikä sille oikeastaan voi mitään. Hakusivun muutokset ovat kuitenkin olleet hyvin harvinaisia, joten hakufunktion todettiin olevan riittävän hyvä.

4.4 Muutokset optimointikäytäntöön

Optimoinnin toiminnan suunnitteleminen ei kuulunut opinnäytetyön sisältöön, mutta suunnitelmat piti kuitenkin ymmärtää, että ne pystyttiin toteuttamaan sovelluksen toimintaan.

4.4.1 Lähtötilanne

Alkuperäinen sovellus oli tehty 10 vuotta sitten ja se ei enää kaikilta osin vastannut todellista tuotantoympäristöä. Alun perin järjestelmään kuului erilaisia optimointitapoja. Kaasumoottorin tehon pystyi määräämään suoraan teho-ohjeena laitokselle tai antamalla laitoslogiikalle ajomallin, minkä perusteella laitoslogiikka osaa säädellä itse moottorin tehoa. Ajo-ohjeita pystyi laskemaan reaaliaikaisesti minuutin välein, kerran tunnissa tai kerran vuorokaudessa.

Optimointialgoritmin tulos ja tuloksen tarkkuus riippuu sähkön pörssihinnoista, jotka muuttuvat joka tunti, optimoitavan kohteen jatkuvasti elävistä sähkö- ja lämpökuormista sekä staattisista, harvoin muuttuvista parametreista, joista oleellisin on maakaasun hinta.

4.4.2 Uudistukset

Optimoinnin toimintaa haluttiin yksinkertaistaa. Ajomallikäytöstä päätettiin luopua, ja siirryttiin pelkästään teho-ohje käyttöön. Päätettiin myös käyttää pelkkää vuorokausioptimointia, koska sen katsottiin olevan nykyään taloudellisesti riittävän tarkka.

Uudessa sovelluksessa päädyttiin seuraavanlaiseen käytäntöön: Kun sähkön hinnat huomiselle vuorokaudelle on julkaistu www-sivulle, ohjelma hakee hinnat automaattisesti ja tallentaa ne tietokantaan. Kun hinnat ovat tietokannassa, laskee ohjelma teho-ohjeet vuorokauden jokaiselle tunnille ja tallentaa ne Access-tiedostoon, josta laitoslogiikan lukuosasovellus siirtää ne laitoslogiikkaan.

Laskemalla teho-ohjeet kerralla koko vuorokaudelle, pystytään muun muassa varmistamaan, että generaattori ei sammu ja käynnisty uudelleen vähän väliä, kuten varsinkin reaaliaikalaskennan käytössä on ollut ongelmana. Lisäksi staattisiin laskentatietoihin lisättiin parametrit, jotka määrittävät minimiajat moottorien käynnissä ja sammuneena olemiselle.

Järjestelmään lisättiin myös parametri, joka arvioi sen ajan, mikä generaattorilla kestää, ennen kuin se rupeaa todellisuudessa tuottamaan sähköä tavoitellulla teholla. Tarkoituksena on että generaattori käy tavoiteteholla koko täyden tunnin.

4.4.3 Ennustekuormat

Vuorokausioptimoinnin toteutuksessa oli ongelmana sähkö- ja lämpökuormien ennustaminen seuraavalle vuorokaudelle. Vanhassa lähdekoodissa oli funktio joka laski kuormaennusteet tietokannasta löytyvien edellisten päivien historiatietojen perusteella, mutta se ei ollut käytössä. Suoritettiin optimoinnin testiajoja InSQL -tietokannasta löytyvien historiatietojen perusteella vertailemalla todellisia arvoja ennustefunktion antamiin arvoihin. Ennusteet todettiin riittävän tarkoiksi nykyisiin olosuhteisiin, joissa kaasun hinta on kuormia suurempi tekijä optimoinnin tuloksen kannalta, joten ennustusfunktio otettiin käyttöön.

4.4.4 Kahden generaattorin tasa-ajo

Toisessa järjestelmän maakaasulaitoksessa on kaksi generaattoria ja aikaisemmin velvoitteena oli ollut pitää toista generaattoria aina käynnissä. Nykyään tällaista velvoitetta ei ole, joten optimoinnin toteutus tältä osin muutettiin.

Kahden generaattorin systeemissä pitää ottaa huomioon kuormien jakaminen generaattoreiden kesken ja varmistaa että kohteen sähkön tarve tulee tyydytetyksi. Sähkön tuotanto voi joissakin olosuhteissa jäädä vajaaksi, jos ensimmäinen generaattori tuot-

taa laitoksen sähkökuormasta liian suuren osan. Tällöin optimointi laskee, ettei toisen generaattorin ajaminen ole taloudellisesti kannattavaa. Tällaista tilannetta varten suunniteltiin ja toteutettiin niin sanottu tasa-ajo systeemi, mikä antaa molemmille generaattoreille saman teho-ohjeen, jos tietyt ehdot täyttyvät.

Koska kahden generaattorin laitoksessa on kuitenkin usein vain yksi generaattori käynnissä kerrallaan, lisättiin ohjelmaan mahdollisuus valita ensisijaisesti ajettava generaattori, jotta molemmat generaattorit kuluisivat tasaisesti.

4.5 Käyttöliittymän toteutus

CHP -optimoija on sovellus, joka suurimman osan aikaa pyörii palvelimella ilman käyttäjän vuorovaikutusta, suorittaen optimoinnin kerran vuorokaudessa. Vaikka ohjelman ajoittainen kaatuminen ei olisikaan katastrofi, oli pääpaino toteutuksessa saada ohjelma toimimaan vakaasti. Käyttöliittymään jäi siis hiottavaa.

Käyttöliittymä toteutettiin Windows Forms -komponenteilla. Käyttöliittymän prototyyppiä ryhdyttiin rakentamaan jo aikaisessa vaiheessa helpottamaan hahmottamaan vanhan sovelluksen toimintaa. Käyttöliittymää suunniteltiin yhdessä opinnäytetyön toimeksiantajan kanssa. Se pohjautuu pitkälti samoihin näyttöihin, mitä alkuperäinen sovellus sisälsi, mutta siitä pyrittiin tekemään selkeäkäyttöisempi ja lisäämään joitakin hyödyllisiä ominaisuuksia.

Käyttöliittymän vaatimuksiin kuului myös mahdollisuus lisätä uusia voimalaitoksia järjestelmään pienellä vaivalla. Tästä johtuen käyttöliittymän komponentit toteutettiin ohjelmallisesti koodissa, eikä käyttäen Visual Studion graafista työkalua.

4.6 Testaus

Sovelluksen osia testattiin prototyyppimenetelmää noudattaen yksi osa kerrallaan. Testausvaiheessa tuli ilmi .NET -sovellukselle melko harvinainen muistivuoto, mikä kaatoi koko ohjelman. Muistivuoto johtui eräästä usein luotavasta käyttöliittymäkomponentista, jota automaattinen roskien kerääjä ei tavoittanut, koska komponenttia ei koskaan varsinaisesti poistettu käytöstä. Ongelma ratkaistiin suunnittelemalla ohjelman kyseinen osa järkevämmin.

4.7 Käyttöohje

Työhön kuului myös käyttöohjeen laatiminen uudistetulle ohjelmalle. Käyttöohjeessa kuvataan optimointijärjestelmän toimintaperiaate ja sovelluksen näytöt, sekä selitetään sovelluksen toiminta käyttäjän kannalta. Lisäksi se sisältää ohjeita ongelmatilanteiden varalle ja ohjeet uuden voimalaitoksen lisäämiseen järjestelmään. Käyttöohje on Haminan Energia Oy:n omistuksessa ja sitä ei katsottu tarpeelliseksi liittää osaksi tätä dokumenttia.

5 UUDISTETUN SOVELLUKSEN TOIMINTA

Järjestelmän toimintaperiaate on edelleen samanlainen kuin ennen, mutta jotkin ominaisuudet ja toiminnot ovat muuttuneet.

Sovellus pyrkii laskemaan voimalaitosten ajo-ohjeet seuraavalle vuorokaudelle, jokaiselle tunnille erikseen. Laskenta tapahtuu sen jälkeen, kun seuraavan vuorokauden sähkön pörssihinnat ovat saapuneet järjestelmän tietokantaan.

Hintojen hakeminen tapahtuu asetuksissa määriteltynä ajankohtana. Jos hintojen haku epäonnistuu, yrittää sovellus hakea hintoja minuutin välein uudestaan. Jos hintoja ei saada haettua lainkaan, ei optimointia voida suorittaa kyseiselle vuorokaudelle ja laitos siirtyy käyttämään laitoslogiikassa olevaa 24 tunnin ajo-ohjetta.

Sovelluksen toiminta on täysin automaattista. Tavallisesti käyttäjän rooliksi jää tulosten tarkkailu ja sovelluksen toiminnan varmistaminen. Staattiset laskentatiedot pitää syöttää käsin, kun niitä on tarve muuttaa.

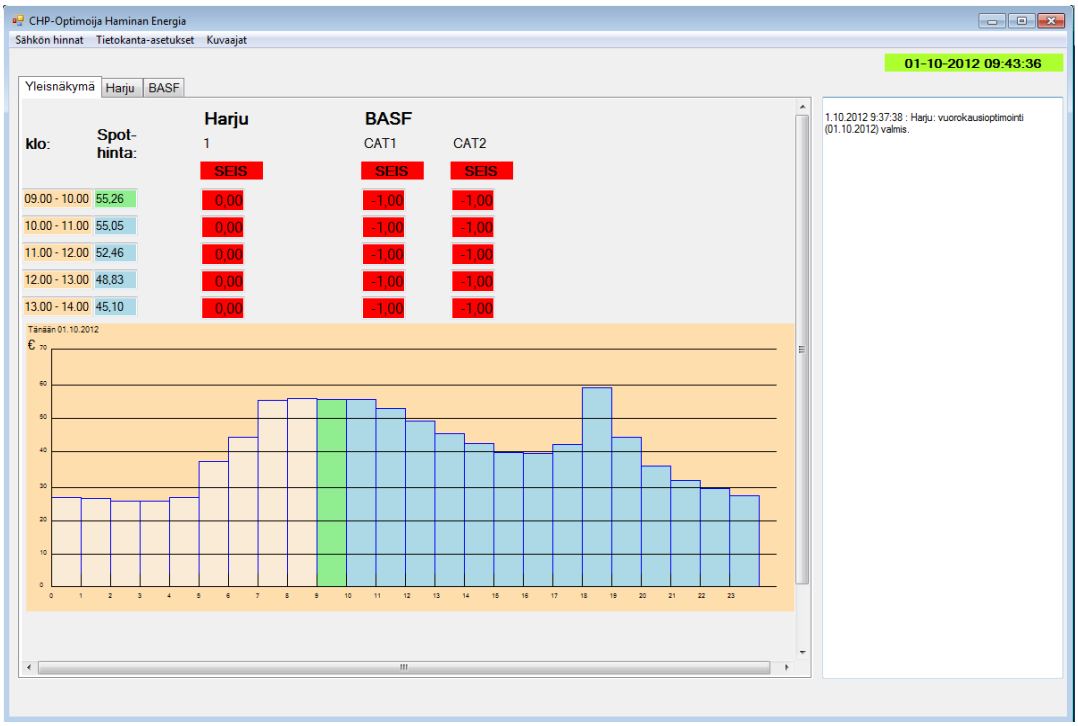
6 KÄYTTÖLIITTYMÄN ESITTELY

Ohjelman käyttöliittymä muodostuu pääikkunasta, Sähkön hinnat-ikkunasta, Tietokanta-asetukset-ikkunasta ja Kuvaaja-ikkunoista.

Pääikkunassa (kuva 4.) on kolme välilehteä: yleisnäkymä ja välilehdet kummallekin järjestelmään liitetylle voimalaitokselle. Ylämenun valikoista pääsee Sähkön hinnat-ikkunaan, Tietokanta-asetukset-ikkunaan ja Kuvaajaa-ikkunoihin.

Pääikkunan oikeassa reunassa on toimintaloki ja sen yläpuolella näkyy päivämäärä ja kellonaika.

Käytön helpottamiseksi on useimpiin käyttöliittymän komponentteihin liitetty selitteen näyttävä ToolTip -komponentti, joka tulee näkyviin, kun hiiri liikutetaan komponentin päälle.



Kuva 4. Pääikkuna

6.1 Yleisnäkymävälilehti

Pääikkunan kuvassa (kuva 4.) näkyvä yleisnäkymä ei sisällä toimintoja, vaan se on ainoastaan tarkoitettu voimalaitosten tämän hetkisen tilan tarkasteluun. Se näyttää taulukossa sähkön pörssihinnat ja voimalaitosten generaattorien tämän hetkisen tilan, sekä ajosuunnitelmat tämän tunnin ja neljän seuraavan tunnin osalta. Värikoodauksen on tarkoitus helpottaa näkemään tilanne. Punainen väri kertoo generaattorin olevan pois päältä (teho-ohje on 0) ja vihreä tarkoittaa generaattorin teho-ohjeen kyseiselle tunnille olevan suurempi kuin 0, siis generaattori on tarkoitus ajaa kyseisenä tuntina. Taulukon alapuolella on pylväsdiagrammi, joka näyttää sähkön pörssihinnat kuluvan vuorokauden osalta.

6.2 Loki

Pääikkunan (kuva 4.) oikeassa reunassa on toimintaloki, mihin päivittyy tekstinä tietoja ohjelman toiminnasta. Päivittyviä tietoja ovat optimoinnin suoristusajankohdat, hintojen hakeminen ja mahdolliset tietokantayhteysvirheilmoitukset.

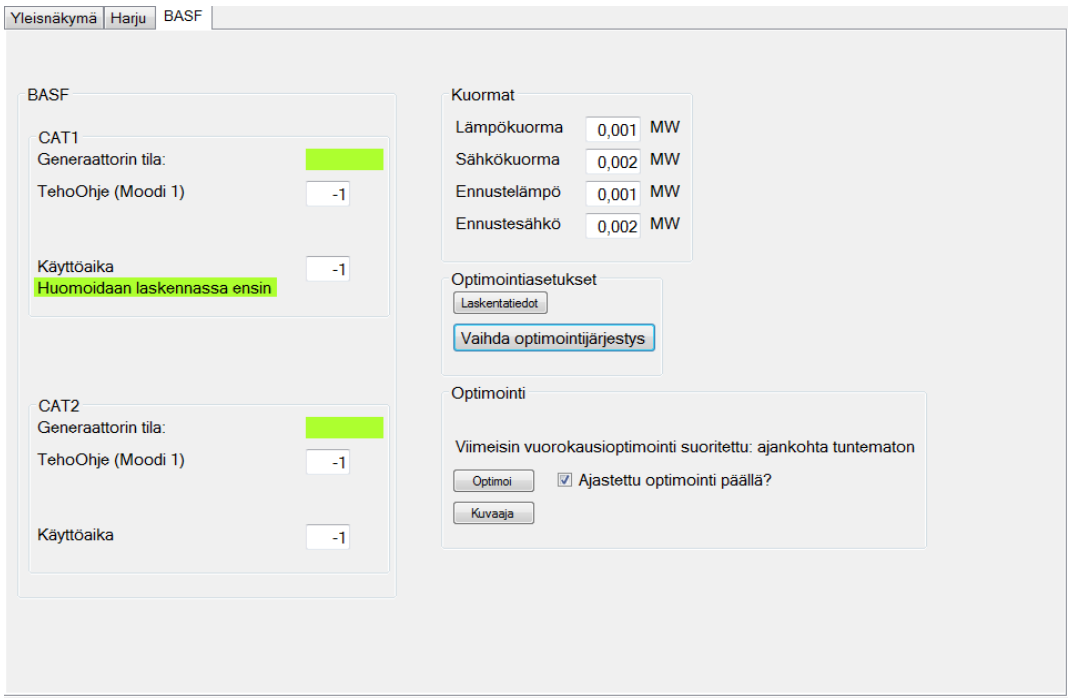
6.3 Voimalaitoskohtainen välilehti

Voimalaitosten optimointia hallitaan niiden omista välilehdistä. Näkymän vasemmassa reunassa näytetään tietoa laitoksen generaattoreista. BASF:lla (kuva.5) on kaksi generaattoria. Generaattorista kerrotut tiedot ovat tämän hetkinen tila (päällä/seis/varavoima) ja teho-ohje. Jos laitoksella on kaksi generaattoria, näytetään myös generaattorin käyttöaika ja ensisijaisuus laskennassa.

Oikeassa reunassa näytetään ylimpänä laitoksen sähkö- ja lämpökuormien 10 minuutin liukuvat keskiarvot. Niiden alla näytetään optimoinnin käyttämät ennustekuormat. Tarkoituksena on antaa mahdollisuus vertailla ennusteiden onnistumista, jos on syytä epäillä ongelmia.

Kuormien alapuolella sijaitsee Laskentatiedot -ikkunan avaava painonappi. Jos voimalaitokseen kuuluu kaksi generaattoria, voi laskennan ensisijaisesti huomioivan generaattorin vaihtaa painonapista, jossa lukee Vaihda optimointijärjestys.

Optimointikehys näyttää ajankohdan, milloin laitoksen optimointi on viimeksi suoritettu. Jos optimointi ei ole suoritettu ohjelman avaamisen jälkeen, on ajankohta tuntematon. Optimoinnin voi myös suorittaa manuaalisesti painamalla Optimoi -painonappia. Automaattisen optimoinnin voi laittaa pois päältä poistamalla rastin viereisestä kentästä. Tavallisesti automaattinen optimointi on kuitenkin aina päällä, ja se suoritetaan joka päivä uusien pörssihintojen saavuttua järjestelmään. Voimalaitoksen tarkemmat ajosuunnitelmat näytetään kuvaajaikkunassa, joka aukeaa Kuvaaja -painonapista.



kuva 5. BASF -välilehti

6.4 Kuvaajaikkuna

Voimalaitoskohtainen kuvaajaikkuna (kuva 6.) on tärkeä osa käyttöliittymää. Se näyttää generaattoreiden ajosuunnitelmat pylväsdiagrammien avulla kuluvan vuorokauden ja seuraavan osalta. Kuvaajat havainnollistavat optimoinnin toimintaa ja laitosten ajosuunnitelmia.

Kuvaajalle saa näkyviin sähkön pörssihinnat, generaattorin ajo-ohjeet ja niiden tuottaman nettohyödyn, sekä tuotantokustannukset. Näkyvissä olevat diagrammit voi valita vasemmassa yläreunassa olevista valintaruuduista. Jos voimalalla on kaksi generaattoria, voi yläreunan radionapeista valita haluaako tarkastella yksittäistä generaattoria vai molempien generaattoreiden yhteisvaikutusta. Yksittäinen tarkastelu kertoo tarkempaa tietoa.

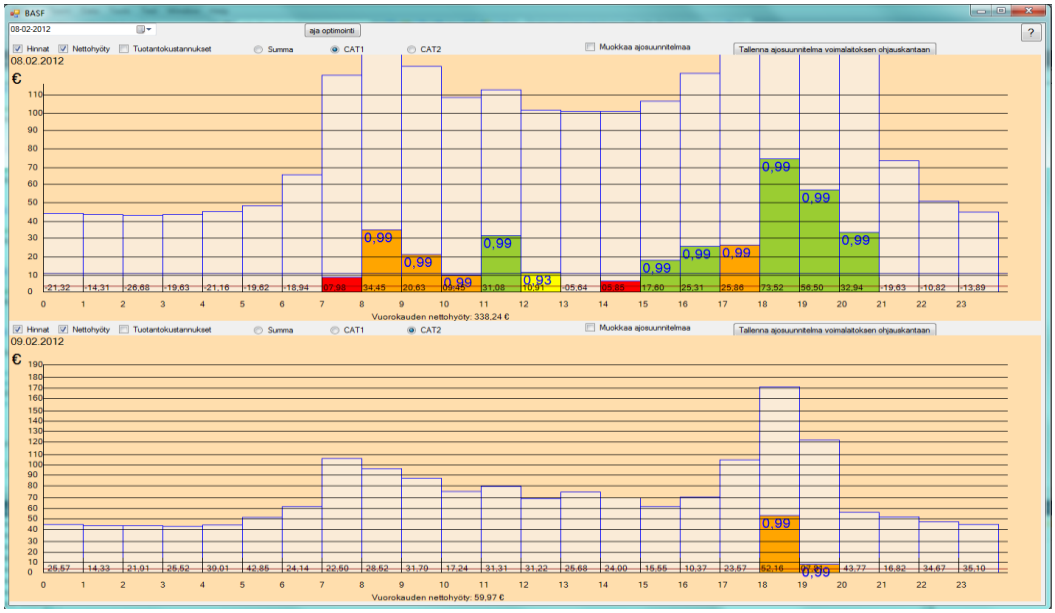
Hinnat ovat yksikköä €/MWh. Nykyinen tunti, menneet tunnit ja tulevat tunnit ovat väritettyinä eri väreillä.

Nettohyöty (€/h) kertoo algoritmin laskeman arvion generaattorin tuottamalle nettohyödylle tunneittain. Koko vuorokauden arvioitu hyöty lukee keskellä alhaalla. Jos tunti on päätetty ajaa, näkyy palkin alaosassa sininen AJO-teksti. Pylvään väri vaihte-

lee riippuen siitä, onko tunti täydellä teholla ajettava, tasa-ajossa, vaajatehoinen vai onko kone seis.

Tuotantokustannukset (€/MWh) näyttävät ajettavien tuntien tuotantokustannukset tunneittain.

Ajosuunnitelmia pääsee muokkaamaan painamalla Muokkaa – painiketta. Muokkaaminen tapahtuu klikkaamalla hiiren vasemmalla painikkeella tulevien tuntien kohdalta. Tunnit asetetaan joko ajettavaksi täydellä teholla tai pois päältä. Tallentaminen tapahtuu painamalla uudelleen samaa painiketta, jossa lukee nyt Tallenna. Toiminnon tarkoituksena on antaa mahdollisuus säätää ajosuunnitelmia manuaalisesti ongelmatilanteissa.



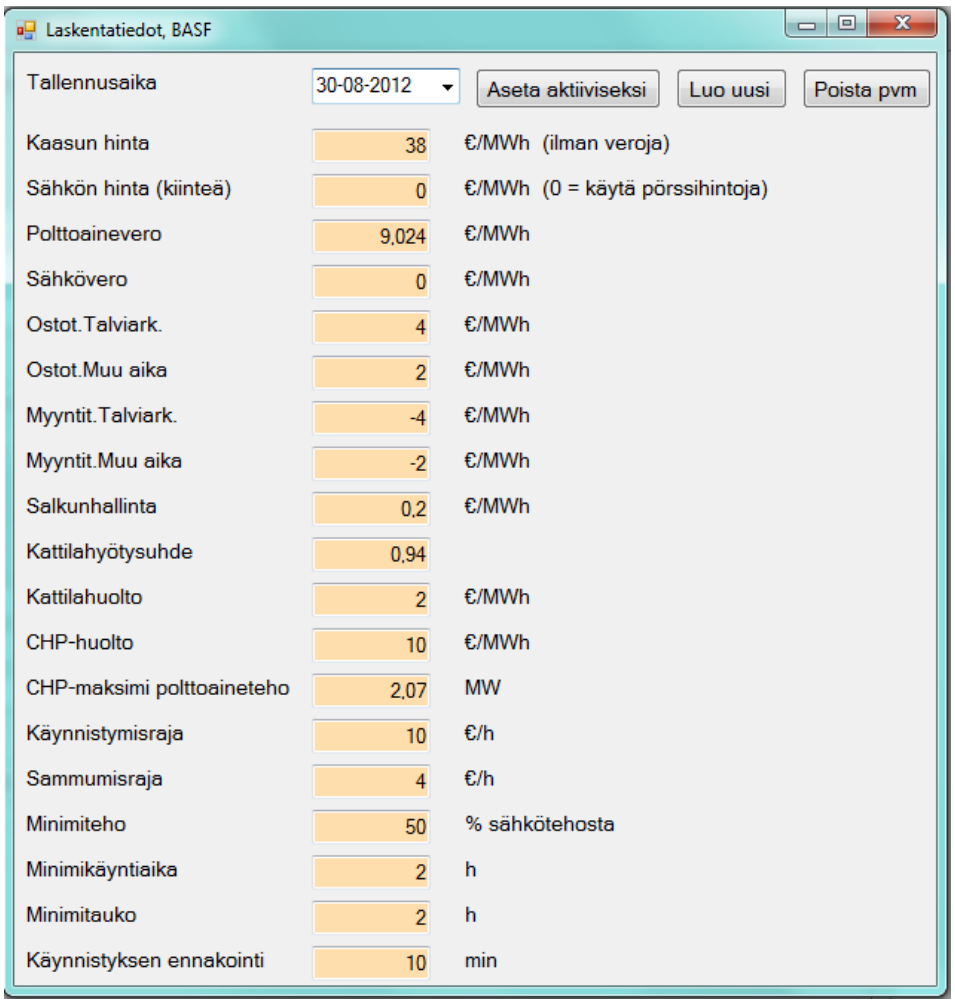
kuva 6. Kuvaajaikkuna BASF

Kuvaajaikkunan ylämenusta aukeaa Värien selitykset -ikkuna (kuva 7.), missä kerrotaan kuvaajan pylväiden värien merkitykset.

kuva 7. Värien selitykset -ikkuna

6.5 Laskentatiedot

Ohjelman käyttämät staattiset lähtötiedot syötetään Laskentatiedot-ikkunan kautta. Valittu päivämäärä voidaan asettaa aktiiviseksi vieressä olevasta painonapista. Aktiivinen päivämäärä on se päivämäärä, minkä alla olevia tietoja ohjelma käyttää laskennassa. Tietoja voi päivittää luomalla uuden päivämäärän Luo uusi-painikkeesta. Lähtötietopäivämäärän voi myös poistaa oikealla olevasta painonapista.



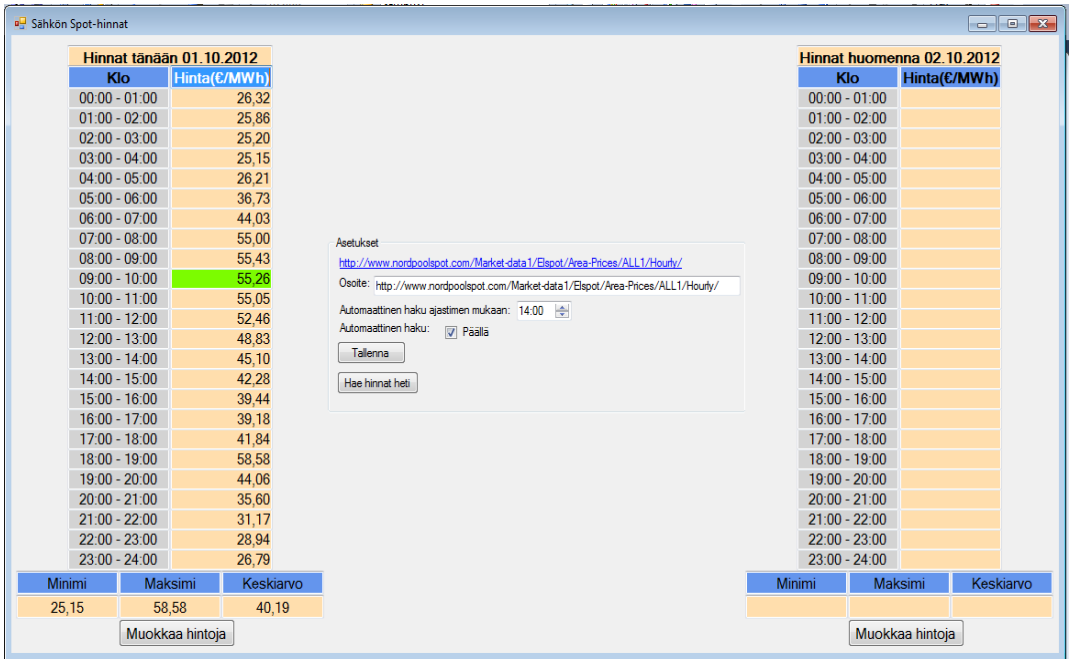
kuva 8. Laskentatiedot-ikkuna

6.6 Sähkön hinnat -ikkuna

Sähkön hinnat -ikkunassa (kuva 9.) voi muuttaa hintojen hakuun liittyviä asetuksia ja tarkastella sähkön pörssihintoja.

Ikkunan vasemmassa reunassa on taulukossa tämän päivän hinnat ja oikeassa reunassa huomisen. Jos hintoja ei löydy tietokannasta, kentät ovat tyhjiä. Hintataulukon alapuolella on Muokkaa hintoja -painike. Sitä painamalla voi kyseessä olevan päivän hinnat syöttää käsin. Tästä on hyötyä esimerkiksi silloin, kun hintojen haku websivulta ei onnistu, tai halutaan testata optimoinnin toimintaa. Syötetyt hinnat tallennetaan tietokantaan painamalla samaa painiketta, jossa lukee nyt Tallenna.

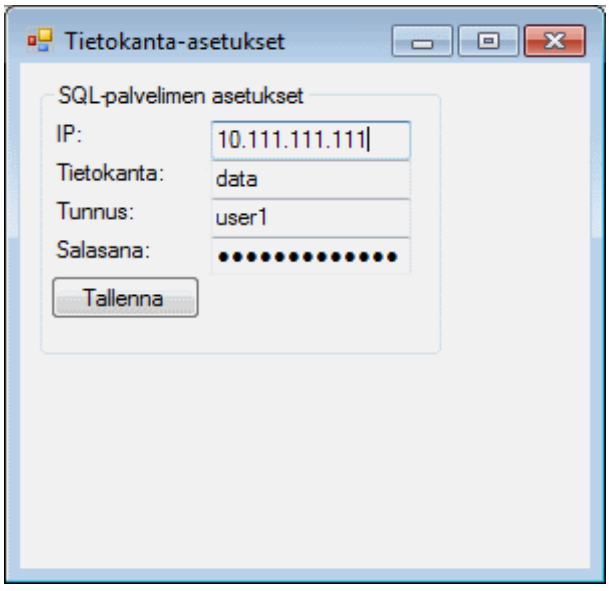
Asetukset sisältävät hakuosoitteen, päivittäisen hakuajankohdan ja mahdollisuuden kytkeä automaattinen haku päälle tai pois päältä. Asetuksiin tehdyt muutokset tulevat voimaan vasta, kun painetaan Tallenna painonapista.



Kuva 9. Sähkön hinnat-ikkuna

6.7 Tietokanta-asetukset

Tietokanta-asetukset-ikkunassa (kuva 10.) pääsee muokkaamaan yhteysasetuksia SQL -palvelimelle. Tiedot syötetään tekstikenttiin ja ne tulevat voimaan, kun painetaan Tallenna -painonappia.



Kuva 10. Tietokanta-asetukset

7 YHTEENVETO JA PÄÄTELMÄT

Opinnäytetyö on opettanut minulle paljon sovelluskehityksestä käytännön tasolla. En ollut aikaisemmin toteuttanut yhtä laajaa ohjelmistokokonaisuutta alusta loppuun. Työssä käyttämäni ohjelmointitekniikat olivat minulle osin tuttuja jo entuudestaan, mutta työn alkuvaiheessa monet asiat olivat silti uusia ja opeteltavaa oli paljon.

Työtä tehdessä tuli erityisesti ilmi dokumentaation ja määrittelyn tärkeys ohjelmistokehitystyössä. Vaikka tarkan vaatimusmäärittelyn tekeminen ei ollutkaan alkuvaiheessa mahdollista, koska ohjelman lopullinen toiminta lyötiin lukkoon vasta melko myöhäisessä vaiheessa, olisi puutteellinenkin määrittely ollut käyttökelpoinen. Määrittely olisi varmasti mahdollistanut järjestelmällisemmän työmenetelmän. Prototyyppimenetelmä sai projektin tuntumaan usein vähän hajanaiselta, eikä selkeän kokonaiskuvan hahmottaminen ollut aina helppoa.

Käyttöliittymään jäi mielestäni paljon parantamisen varaa. Erityisesti ohjelman käytettävyyden miettiminen ja testaaminen jäi liian vähälle huomiolle, vaikka se ei työn pääpainopiste ollutkaan.

Työn tekeminen oli kuitenkin mielekästä ja koin opinnäytetyön aihepiirin mielenkiintoiseksi. Voin olla tyytyväinen lopputuloksena saatuun sovellukseen, joka täyttää sille asetetut vaatimukset ja on otettu käyttöön vanhan sovelluksen tilalle.

LÄHTEET

1.Tulokas, T. 2002.Optimointijärjestelmän kehittäminen sähköä ja lämpöä tuottavaan voimalaitokseen. Saatavilla: http://www.doria.fi/handle/10024/29827 [viitattu 16.4.2013].

2.Getting Started with the Industrial SQL Server Historian.2006. http://trainweb.wonderware.com/getstartinsql/InSQLPages/aM01_Overview.htm [viitattu 18.4.2013].

3. Getting Started with the Industrial SQL Server Historian.2006. http://trainweb.wonderware.com/getstartinsql/InSQLPages/aM03_Overview.htm [viitattu 18.4.2013].

4. Haukilehto, A. 2002.Visual C#.NET. Helsinki. Edita Publishing Oy.

5. Microsoft. 2013. Microsoft Developer Network. http://msdn.microsoft.com/en-us/library/8bxxy49h.aspx[viitattu 16.4.2013].

6. Microsoft. 2013. Microsoft Developer Network. http://msdn.microsoft.com/en-us/library/0h5y8567.aspx[viitattu 16.4.2013].

7. Microsoft. 2013. Microsoft Developer Network. http://msdn.microsoft.com/en-us/library/dfty2w4e.aspx[viitattu 16.4.2013].

8. Microsoft. 2013. Microsoft Developer Network. http://msdn.microsoft.com/en-us/library/windows/desktop/ms718124(v=vs.85).aspx [viitattu 16.4.2013].

9. Microsoft. 2013. Microsoft Developer Network.http://msdn.microsoft.com/en-us/library/windows/desktop/ms723100(v=vs.85).aspx [viitattu 16.4.2013].

10. Html Agility Pack.2012. http://htmlagilitypack.codeplex.com[viitattu 16.4.2013].

11. Microsoft. 2013. Microsoft Developer Network. http://msdn.microsoft.com/en-us/vstudio/cc136611.aspx[viitattu 16.4.2013].

12 Microsoft. 2013. Microsoft Developer Network. http://msdn.microsoft.com/en-us/library/vstudio/dd492132.aspx[viitattu 18.4.2013].

13. Haikala,I & Mikkonen,T. 2011.Ohjelmistotuotannon käytännöt. Hämeenlinna. Talentum Media Oy.

14. Gunderloy, J. & Caison, M. 2002. Mastering Visual C# .NET. Alameda, CA, USA,Sybex, s.830. Saatavilla: http://www.kyamk.fi/kirjasto[viitattu 20.4.2013].